1 TOMCAT

1.1 tomcat下载和安装

1.1.1 下载

下载地址：http://tomcat.apache.org/
有解压版 和 安装版，还分windows 和 linux版，根据自己的需求，选择对应的版本下载.
tomcat服务器运行需要jdk的支持，版本对应为：

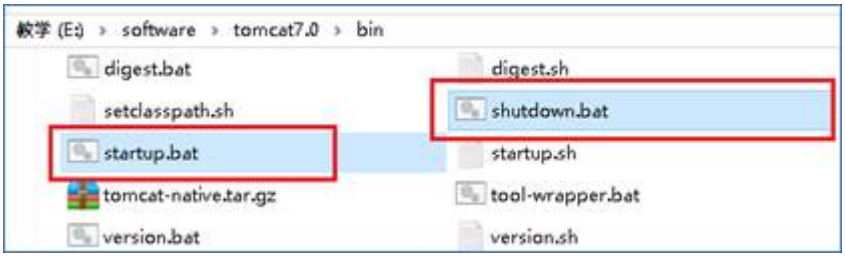
tomcat5 需要jdk4以上支持
tomcat6 需要jdk5以上支持
tomcat7 需要jdk6以上支持
tomcat8 需要jdk7以上支持

1.1.2 安装及启动

（1）安装
绿色版解压之后就可以使用

解压后还需要配置JAVA_HOME环境变量，指向jdk的根目录，指定tomcat启动时使用哪个jdk

（2）启动
通过bin/startup.bat启动tomcat
通过bin/shutdown.bat关闭tomcat



测试: 通过访问 http://localhost:8080 如果能够看到tomcat的首页就证明tomcat安装配置成功

注意：tomcat安装路径中一定不能有中文和空格，可能一时半会没问题，但是不知道什么时候就可能出现意外。

1.2 Tomcat配置

1.2.1 修改默认的端口号

tomcat服务器默认监听的端口号为 8080，每次访问时都需要在主机名或IP地址后跟上端口号，如果想省略不写，将端口号修改为80即可!

找到 [tomcat]/conf/server.xml文件(tomcat服务器的核心配置文件)，修改文件中(70行) 的如下配置:

```
<Connector port="80" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

将port改为80即可!!

1.2.2 端口占用问题

如果在启动时报端口占用的错误：

```
java.net.BindException: Address already in use: JVM_Bind <null>:80
        at org.apache.tomcat.util.net.JIoEndpoint.bind(JIoEndpoint.java:
        at org.apache.tomcat.util.net.AbstractEndpoint.init(AbstractEndp
```

是因为服务器在启动的过程中监听80端口，而该端口已经被别的进程所占用，因此服务器启动失败!

解决方式一：找到**shutdown.bat**命令，双击运行，将服务器按照正常的流程再关闭一次!

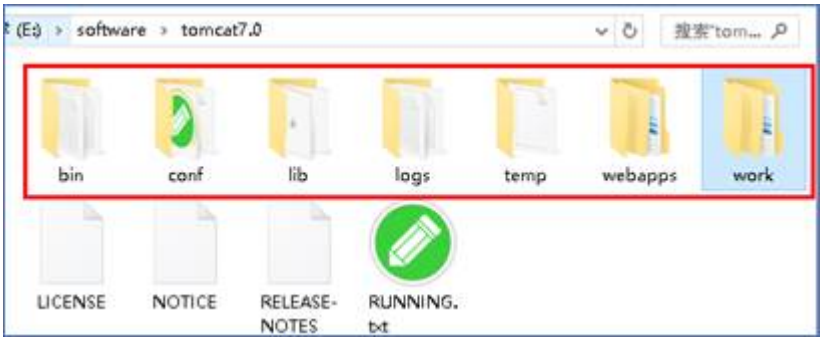
解决方式二：如果是别的程序占用了80端口,导致服务器启动失败，在cmd中通过**netstat -ano**命令，查看占用80端口的进程，例如：

```
\Users\sz>netstat -ano

动连接

协议   本地地址          外部地址          状态           PID
TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       3548
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       992
```

根据进程ID找到进程，结束即可。

也可以利用命令**taskkill /pid 进程ID**来杀死指定ID的进程

1.3 tomcat的目录结构



bin：tomcat批处理文件的存放目录

conf：tomcat配置文件所在的目录，其中server.xml是tomcat的核心配置文件

lib：tomcat运行时所依赖的jar包存放的目录.

logs：tomcat日志文件所在的目录

temp：tomcat运行时产生的临时文件存放的目录

webapps：是localhost虚拟主机管理的目录，放在这个目录下的web应用可以通过浏览器访问localhost主机来访问

work：tomcat运行时产生的工作文件存放目录. 是tomcat的工作目录

1.4 web应用

1.4.1 什么是WEB应用

将为了实现某一功能而准备好的所有的web资源按照一定的目录结构组织起来的就是一个web应用

虚拟主机不能直接管理web资源，web资源必须组织成web应用才可以交给虚拟主机去管理

1.4.2 web应用的目录结构

```
game
|
|-- 静态web资源，jsp 直接放在web应用的根目录下，可以通过浏览器直接访问
|-- WEB-INF web应用中特殊的目录，这个目录可以没有，一旦有了就必须符合一定的目录结构
        |   (放在这个目录中的资源将被保护起来，通过浏览器不能直接访问)
         |--classes 用来存放动态web资源的class文件
         |--lib 用来存放动态web资源所依赖的一些jar包
         |--web.xml 当前web应用的核心的配置文件，web应用所有的配置操作都需要在这个文件中进行
```

1.4.3 部署web应用到虚拟主机中

在tomcat服务器中提供了一个虚拟主机：localhost

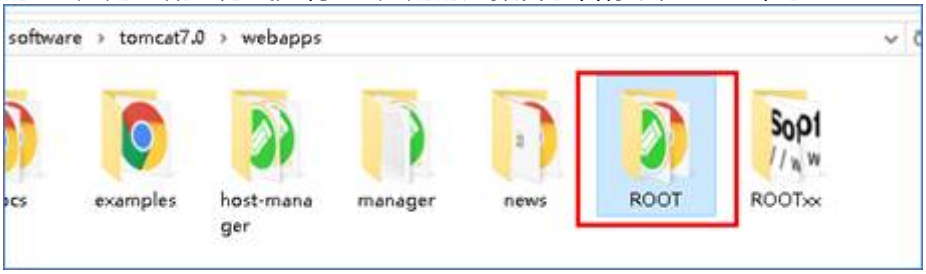
```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

因此我们可以将WEB应用部署在localhost主机下.

部署方式：将组织好的WEB应用的目录直接丢进localhost主机默认管理的目录下(webapps)即可，这种配置方式不需要重启服务器就可以起作用!

1.4.4 配置缺省的(默认的)WEB应用

在访问服务器中的WEB应用下的资源时，如果不想写WEB应用的虚拟目录的名称，可以将当前WEB应用配置为缺省的WEB应用. 配置方式是将WEB应用的虚拟目录名称改为ROOT即可!!

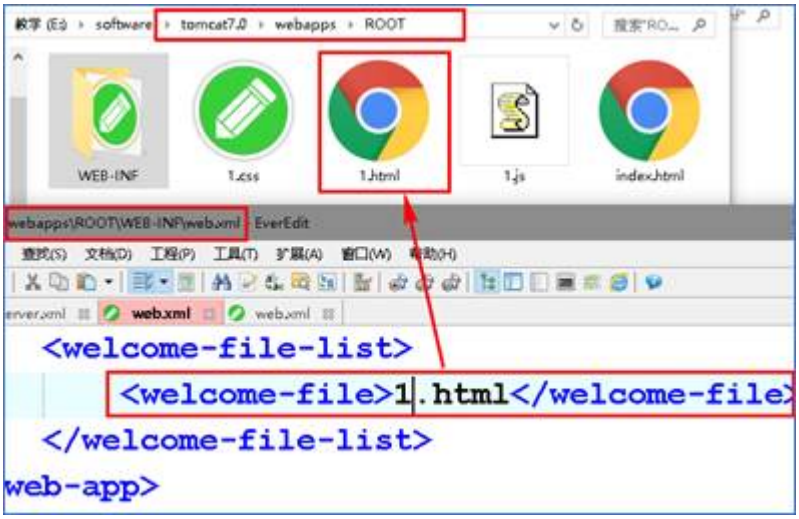


1.4.5 配置WEB应用的主页

如果在访问WEB应用下某一个资源时(比如1.html)，不想书写资源的路径，可以将这个资源页配置为WEB应用的主页，在访问时就可以省略该资源的路径。

配置方式为：在WEB应用的web.xml文件中，添加如下配置：

```
    <welcome-file-list>
        <welcome-file>1.html</welcome-file>
    </welcome-file-list>
</web-app>
```



1.5 虚拟主机

1.5.1 什么是虚拟主机

所谓的虚拟主机就是tomcat服务器中配置的一个站点,在访问时就好像访问一台真实的主机一样
tomcat服务器中可以配置多个站点，一个站点就是一台虚拟主机

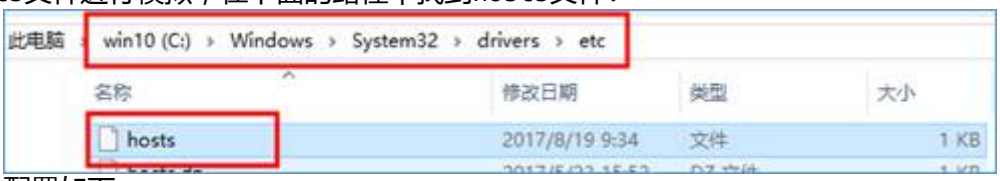
1.5.2 配置虚拟主机

在[tomcat]/conf/server.xml中的server/service/Engine标签内部添加一个Host标签:

```
<!-- Host setting -->
<Host name="www.baidu.com" appBase="baidu"></Host>

</Engine>
</Service>
</Server>
```

Host标签上的name是必须存在的属性，用来指定虚拟主机的名称.
Host标签上的appBase是可选属性，用来指定虚拟主机默认管理的目录，如果没有配置该属性，表示当前主机没有默认管理的目录!

配置完后，还需要在DNS服务器中配置主机名和IP地址的映射关系，但是DNS服务器一般不能修改，可以通过hosts文件进行模拟，在下面的路径中找到hosts文件:



配置如下:

1.5.3 配置缺省的（默认的）虚拟主机

如果通过主机名来访问，访问的就是对应的主机. 如果通过IP地址来访问，服务器不知道你访问的是哪一台虚拟主机，这时将会访问缺省的虚拟主机。缺省的虚拟主机配置如下：(默认是localhost)

```
<Engine name="Catalina" defaultHost="localhost">
```

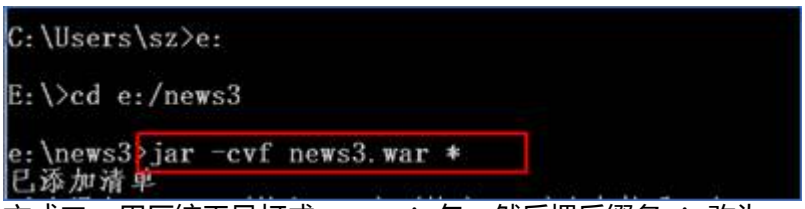
1.5.4 综合练习

配置为www.163.com虚拟主机，部署music web应用，将WEB应用配置为缺省web应用，并且配置web应用的主页，最终实现直接访问www.163.com能够显示主页的内容.

1.6 其他相关

1.6.1 打war包

方式一: 进入应用的目录，用 jar -cvf xxx.war * 命令，就可以将当前目录下的所有内容打成war包，例如：



方式二：用压缩工具打成 xxx.zip包，然后把后缀名zip改为war即可

2 HTTP协议

2.1 什么是HTTP协议?

用来规定浏览器客户端和服务器之间进行通信的方式

2.2 三个基本原则

- 基于请求响应模型
- 一次请求对应一次响应
- 请求只能由客户端发出，服务器只能被动的等待请求，做出响应.

2.3 HTTP协议详解

2.3.1 HTTP请求

1.请求行(包含了请求方式、请求资源的路径、遵循的协议及版本)

```
GET /news3/1.html HTTP/1.1
```

GET：请求方式，在HTTP协议中一共规定了7种请求方式， 只用GET和POST

/news3/1.html：请求资源的路径

HTTP/1.1：浏览器发送请求时所使用的协议及版本

2.若干请求头

http协议中请求头非常多，下面列出常见的请求头及其功能：

```
Accept: text/html,image/*
-- 通知服务器当前浏览器可以接受那些格式的数据
Accept-Charset: ISO-8859-1
-- 浏览器可以接受的字符集编码
Accept-Encoding: gzip,compress
-- 浏览器可以接受的压缩格式
!Host: www.tedu.cn:80
--  需要访问的虚拟主机的名称
!!Referer: http://www.tedu.cn/index.jsp
-- 这是和防盗链相关的头,对当前资源的访问来自哪个页面的超链接
```

User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0) -- 客户端的基本信息

!!!Cookie（后面讲）
-- 和cookie相关的头
Connection: close/Keep-Alive
-- 是否继续保持连接
Date: Fri, 17 Feb 2017 18:23:51 GMT
-- 当前发送请求的时间

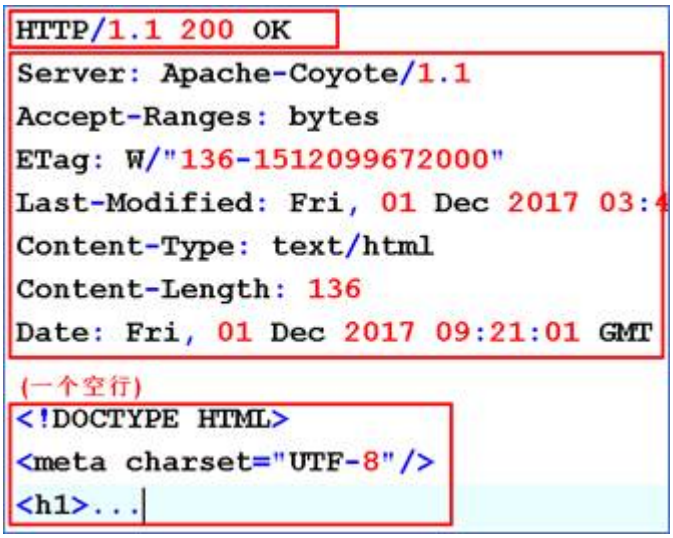
3.请求实体内容

如果请求方式是GET提交，请求实体中没有数据
只有当请求方式为POST提交，并且请求中携带了数据， 请求实体才会有内容

4.GET请求和POST请求方式的区别

主要体现在请求参数发送过程的不相同
GET提交：通过请求行拼接参数将数据发送给服务器
（1）通过地址栏携带参数，非常不安全
（2）通过地址栏发送数据，数据量不能太大(不能超过1kb或者是4kb);
POST提交：通过请求实体内容携带参数，数据不会显示在地址栏
（1）参数不会出现地址栏，相对更安全
（2）数据通过请求实体内容发送，数据量理论上没有限制.

2.3.2 HTTP响应



1.状态行

HTTP/1.1 200 OK

HTTP/1.1：服务器做出响应时遵循的协议及版本
200：状态码（一个三位的数字）， 表示服务器处理请求的结果如何，200表示服务器成功的处理请求
- 200: 服务器成功的处理了请求
- 302: 和location响应头配合实现请求重定向.
- 304: 表示通知浏览器使用本地缓存
- 404: 表示客户端请求的资源不存在!
- 500: 表示服务器端发生了错误!

OK：描述短语， 也是用来表示服务器处理请求的结果。

2.响应头

http协议中响应头头非常多，下面列出常见的响应头及其功能：
!!!Location: http://www.tedu.cn/index.jsp
-- 配合302使用实现请求重定向
Server:apache tomcat
-- 服务器的基本信息

Content-Encoding: gzip
-- 服务器发送的数据使用的压缩格式
Content-Length: 80
-- 服务器发送的数据的大小
!!!Content-Type: text/html; charset=GB2312
-- 服务器发送的数据是什么格式的,如果是字符格式的数据,则还可以通知服务器发送的数据使用的是什么编码,浏览器会根据这个头指定的编码决定使用哪个编码来打开收到的数据

!!Refresh: 1;url=http://www.tedu.cn
-- 定时刷新相关的头,通知浏览器,过几秒后自动刷新访问哪个地址
Content-Disposition: attachment;filename=aaa.zip
-- 通知浏览器以附件的形式打开发送过去的数据,是和文件下载相关的头
!!!Set-Cookie:SS=Q0=5Lb_nQ; path=/search
-- 和Cookie相关的头
Expires: -1
-- 通知浏览器是否缓存
Cache-Control: no-cache
-- 通知浏览器是否缓存
Pragma: no-cache
-- 通知浏览器是否缓存
-- 之所以一个功能有三个头,是因为历史原因.所以为了网页的兼容性考虑,通常这三个头要一起使用
Connection: close/Keep-Alive
-- 是否保持连接
Date: Fri, 17 Feb 2017 18:24:11 GMT
-- 响应时的时间

3.响应实体内容